REMARKS

Claims 1-22 were pending in the subject case. In the Office Action dated 5/8/02 ("Office Action), Claims 1-7, 11-15, and 19-22 were rejected, and 8-10 and 16-18 were

5   objected to. Claims 1-22 remain pending. In the amendment set forth above, Claims 1, and 11-18 are amended, and the remaining Claims are as previously presented. In view of the amendments to the claims and the arguments set forth below, it is respectfully submitted the Claims are in condition for allowance.

10

ARGUMENTS

1.   A replacement Figure 10 is submitted herewith, including the label "354" for part 354, which is labeled in red ink. It will be noted that in the originally filed copy of Figure 10, only

15   the label "354", and not the part itself, is mistakenly truncated. With this change to Figure 10, it is respectfully submitted that this objection should be withdrawn.

2.   The Abstract of the disclosure is objected to because it exceeds the maximum allowable length.   A new Abstract is submitted herewith that conforms to length

20   requirements, and it is therefore submitted that this objection should be withdrawn.

3.   The Examiner asserts that the title of the invention is not description. A new title is provided above that even more clearly describes the invention. It is respectfully submitted that this objection should be withdrawn.

25

4.   Claims 11-18 are rejected under 35 USC §112, second paragraph, as being indefinite. More specifically, the Examiner objects to the use of the term "pipeline depth controller" in these Claims. Each of Claims 11-18 has been amended to recite a "pipeline controller" rather than a "pipeline depth controller". Aspects of this controller are recited in

30   the various elements of the Claims. With this change, it is believed that Claims 11-18 conform to the requirements of 35 USC §112, second paragraph, and this rejection should be withdrawn.

5.   Claims 1, 11, 13, and 19 are rejected under 35 USC §102(e) as being anticipated by U.S. Patent No. 5,996,064 to Zaidi et al. ("Zaidi"). This rejection is respectfully traversed.

Claim 1, as amended, claims a first storage device to store a programmable count value *indicative of a predetermined number of instructions*, and a logic sequencer to initiate concurrent execution on the predetermined number of the instructions in the predetermined period of time.   Claim 1 therefore claims the aspect of Applicants' invention wherein a predetermined programmable value indicates the number of instructions that will enter the pipeline during the predetermined period of time, where that predetermined period of time is defined by the number of stages in the pipeline.

In contrast to Applicants' Claim 1, Zaidi teaches defining "post-ready latency" values for each instruction. An instruction is scheduled for execution so that the instruction follows an earlier instruction by an amount of time at least equal to the post-ready latency value for that instruction. (Zaidi column 6 lines 1-6.) Thus, Zaidi controls the execution of instructions by defining *delay values* that determine how must time must elapse between two instructions, rather than by defining the number of instructions that will begin execution within the instruction pipeline within a given time period. The different between Applicants' and the Zaidi approaches is summarized in Applicants' Specification in the description of the prior art as follows:

"A system for controlling pipeline execution in a programmable manner is described in U.S. Patent No. 5,911,083 entitled "Programmable Processor Execution Rate Controller" to Kuslak, which is assigned to the assignee of the current invention. This patent describes a system *for preventing additional instructions from entering the instruction pipeline for a selected amount of time after selected ones of the instructions enter the instruction pipeline.* This may be referred to as "de-piping" the pipeline. Because additional instructions are not entering the pipeline as the execution of the resident instructions is completing, certain timing conflicts can be avoided. Additionally, this mechanism can be used to control the execution rate of the processor, if desired.

Although the prior art system is capable of selectably de-piping the instruction pipeline, this mechanism is only selectable on an instruction-by-instruction basis, and is not controllable based on selectable instruction combinations. In other words, if a particular instruction is selected to trigger the de-piping mechanism, the de-piping occurs every time the instruction enters the instruction pipeline, instead of merely for those combinations of instructions that result in timing conflicts. This slows processor execution unnecessarily in those instances where the processor is de-piped when no conflict actually existed.   Additionally, the described de-piping mechanism is not responsive to system conditions. That is, the triggering of the de-piping mechanism can not be controlled based on the occurrence of such system conditions as errors or interrupts. Finally, the de-piping mechanism can not be used to efficiently solve timing conflicts that are caused by two non-contiguous instructions within the instruction

stream, or that are caused by a combination of more than two instructions. This is because the prior art de-piping mechanism inserts delay into the pipeline immediately following a particular instruction instead of controlling the number of instructions that are concurrently executing within the pipeline." (Applicants' Specification page 5 lines 6-28.)

5   As can be appreciated from the foregoing excerpt, Applicants' Specification compares Applicants' invention to a prior art system that is similar to the Zaidi system. That is, the discussed prior art system utilizes defined delays between instructions to control instruction execution. Applicants' Specification sets forth the various advantages of Applicants' system

10   as compared to prior art systems such as that of Zaidi. These advantages include the ability to address timing conflicts that are caused by two non-contiguous instructions within the instruction stream, and the ability to guarantee that only a predetermined number of instructions are resident within the pipeline at a given time. The Zaidi system cannot control the number of instructions that are undergoing concurrent execution, as can be appreciated by the following

15   examples:

Assume that in the Zaidi system, post-ready latency values of 1, 2, and 3 cycles are defined for instructions N1, N2, and N3, respectively. In a system having a pipeline that has four stages to accommodate, at most, four instructions, the number of instructions within the pipeline at any given time will depend entirely on the ordering of the instructions N1, N2, and

20   N3 within the instruction stream. For example, if the sequence of instructions "N1, N2, and N3" is presented to the Zaidi IP for execution, only two instructions are resident within the pipeline at a given time based on the defined latency values. If, however, the sequence is "N3, N2, and N1", all three instructions are resident within the pipeline at the same time. Thus, the Zaidi system cannot control the *number* of instructions that begins execution in a

25   predetermined period of time, but instead can only control the *delay between two consecutive instructions.*

It may be further noted that even if the Zaidi system defines all latency values to be the same for every instruction, it would still not result in controlling the number of instructions that begin simultaneous execution within a given period of time. This is because not all

30   instructions execute in the same number of clock cycles, and further because error and interrupt conditions affect how instructions begin and continue execution. This may be appreciated by considering Applicants' Figures 12 and 15. In both Figures 12 and 15, Applicants' controller "de-pipes" the pipeline so that only three instructions begin execution during a predetermined period of time. However, the delays that must be inserted between

35   the instructions to cause this to happen differ in the two examples. This is because Figure

12 involves a case wherein all "standard" instructions are executing, whereas Figure 15 depicts an instance involving execution of an "extended mode" instruction that requires additional cycles. Thus, even if every latency value was defined as being the same (e.g., 2 cycles) for every instruction in the instruction set, the Zaidi system is still not capable of

5   controlling the *number* of instructions that begin execution in a predetermined time period.

Because the Zaidi system utilizes defined *delays between consecutive instructions* rather than a programmable count value that specifies the *number of instructions that will begin execution within a predetermined time period*, Zaidi does not anticipate Applicants' invention. Moreover, Applicants' system is more than an obvious modification of Zaidi.

10   Applicants' invention requires the use of complex timing sequences as shown in Figures 10-17 to implement a system having important advantages over the Zaidi mechanism. For at least this reasons, Claim 1 is allowable over Zaidi as presently presented, and this rejection should be withdrawn.

Turning now to a discussion of Claim **11**, this method Claim includes the steps of

15   providing a count to the pipeline controller, and utilizing the pipeline controller to initiate the execution of the number of instructions specified by the count within the predetermined period of time. As discussed above, Zaidi does not disclose use of a count that specifies the *number of instructions to begin execution* in a predetermined period of time. For reasons similar to those discussed above with respect to Claim 1, Claim 11 is not anticipated by Zaidi,

20   and this Claim is allowable over this rejection.

Claim 13 depends from Claim 11 and is allowable over the current rejection for at least the reasons discussed in the foregoing paragraph. Additionally, Claim 13 describes storing respective count signals for each of predetermined ones of the instructions, wherein the count signals control the number of instructions that enters the pipeline in a

25   predetermined period of time. This additional aspect is not taught by Zaidi, which teaches storing latency values for instructions, wherein the latency values *do not control* the number of instructions that enters the pipeline in a predetermined period of time. For this additional reason, Claim 13 is allowable over this rejection, which should be withdrawn.

Claim 19 claims sequencer means for controlling the entry of instructions into the

30   instruction pipeline such that concurrent execution is initiated for *the number of instructions specified by programmable count signals* within a predetermined period of time. As noted above, Zaidi does not teach the use of any programmable signals that specify the number of instructions that will begin execution within a predetermined period of time. For at least this

reason, Zaidi does not teach Applicants' invention of Claim 19, and this rejection should be withdrawn.

To summarize, based on at least the reasons set forth above, Zaidi does not anticipate Claims 1, 11, 13, and 19, and this rejection should be withdrawn.

5

6.   Claims 2, 5, 6, 12, 14, and 20 are rejected under 35 USC §103(a) as being unpatentable over Zaidi in view of U.S. Patent No. 6,209,083 to Naini et al. ("Naini"). This rejection is respectfully traversed.

Claims 2, and 5-6 depend from Claim 1 and are allowable over this rejection for at

10   least the reasons discussed above with respect to Claim 1. It will be noted that Naini does not add anything to Zaidi that would teach a programmable count that indicates the number of instructions that begins execution during a predetermined time period.

These Claims include additional aspects that are not taught or suggested by Zaidi or Naini, alone or in combination. For example, Claim 5 claims programmably enabling the

15   logic sequencer to repeatedly generate a pipeline control signal so that execution of the predetermined number of instruction is initiated during successive periods of time equal to the predetermined period of time. This mode is discussed on page 8, first full paragraph, of Applicants' Specification. The Examiner states that Zaidi and Naini teach this aspect of the invention because Zaidi always operates using the latency values, and further because Naini

20   teaches the use of different modes. Applicants' Representative disagrees with this assessment. Neither Zaidi nor Naini teach the use of a programmable "repeat" mode that may be selected to allow a pipeline controller to repeatedly execute in a particular manner during every successive predetermined time period (instead of operating on a "single-shot" basis.) More specifically, Zaidi does not teach the use of a programmable mode at all, and

25   the mode switch taught by Naini does not in any way suggest programmably enabling repetitive operation. For this additional reason, Claim 5 is allowable over this rejection.

Claim 6 includes the limitation of having programmable enable logic to enable the logic sequencer to receive programmable count values for first selectable ones of the instructions. The Examiner states that this is taught by the Zaidi/Naini combination. Again,

30   Applicants' Representative disagrees. Any control performed by the Zaidi system appear to be *perpetually* functioning and cannot be programmably enabled via any type of selectable mode. Moreover, the mode switch of Naini does not teach programmable logic to enable use of count values applying exclusively to *first selectable ones* of the instructions. The mode

switch in Naini appears to select whether *all instructions* will execute in normal mode, or *all instruction* will execute in a fast mode, and does not distinguish among any set of first instructions, as claimed by Applicants' Claim 6. For at least this additional reason, Claim 6 is allowable over this rejection.

5   Claims 12 and 14 depend from Claim 11 and are allowable over this rejection for at least the reasons discussed above with respect to Claim 11. These Claim includes additional aspects not taught or suggested by the cited combination of references. For example, Claim 12, claims a limitation similar to that discussed above in reference to Claim 5. For the additional reasons discussed above in reference to that Claim, Claim 12 is

10   allowable over this rejection. Claim 14 claims a limitation similar to that discussed above in reference to Claim 6, and is allowable over the cited combination of references for the additional reasons discussed above in reference to Claim 6.

        Claim 20 depends from Claim 19 and is allowable over this rejection for the reasons discussed above with respect to Claim 19. Claim 20 further includes a limitation similar to

15   that discussed above in reference to Claim 5, and is allowable over this rejection for the reasons discussed above in regards to Claim 5.

        Finally, there is no motivation to combine the cited references. The Zaidi system relates to a method of scheduling instructions in an out-of-order processor. (Zaidi column 6 lines 66-67.) In contrast, Naini describes a system and method for providing selectable

20   exception handling modes for an in-order instruction processor. (See, for example, Naini title, and Naini column 8, lines 22-53, which describes that instructions proceed through the pipeline as they appear within the instruction stream, and as they are stored within instruction queue 364.) There would be no motivation to combine aspects of the Naini exception handling system for an in-order instruction processor with the Zaidi instruction scheduler for

25   an out-of-order processor. For this additional reason, this rejection is improper and should be withdrawn.

        In sum, Claims 2, 5, 6, 12, 14, and 20 are allowable for reasons discussed above in reference to the base Claims. Further, these Claims includes additional limitations not taught or suggested by the cited combination of references, and are allowable over this rejection ,

30   which is improper, and should be withdrawn.

7.   Claims 3, 4, 7, 15, 21, and 22 are rejected under 35 USC §103(a) as being unpatentable over Zaidi in view of Naini and further in view of U.S. Patent No. 5,475,824 to Grochowski et al. ("Grochowski"). This rejection is respectfully traversed.

Claims 3, 4, and 7 depend from Claim 1 and are allowable over this rejection for at least the reasons discussed above in reference to Claim 1. It may be noted that Grochowski does not add anything to the Zaidi/Naini combination that would teach or suggest using a count value to specify the number of instructions that begins execution within a pipeline in a predetermined period of time.

Claim 3 claims the limitation of enabling the logic sequencer to de-pipe the pipeline when a predetermined *combination* of instructions enters the pipeline. The Examiner states that this aspect of the invention is taught by Grochowski. Grochowski teaches detecting whether two sequential instructions have dependencies, and if not, issuing them in parallel using two separate pipelines to thereby *increase* the parallelism associated with instruction execution. Grochowski does not, therefore, teach or suggest Applicants' invention of detecting instruction combinations that will be used to *decrease* the parallelism of instruction execution by selectively de-piping of the pipeline. To re-state, the detection of an instruction combination in Grochowski results in more parallelism, whereas the detection of an instruction combination as described in Applicants' Claim 3 results in less parallelism. For at least this reason, Grochowski actually *teaches away* from Applicants' invention. For at least this additional reason, Claim 3 is allowable over this rejection, which should be withdrawn.

Moreover, for reasons similar to those discussed in the foregoing paragraph, the cited combination of references is improper. As discussed above, there is no motivation to combine the exception handling system of the Naini in-order instruction processor with the instruction scheduler for the Zaidi out-of-order instruction processor. Additionally, there is no motivation to combine the Grochowski system with Zaidi. As previously noted, Grochowski teaches a system for increasing instruction execution parallelism by issuing multiple instructions *at once* using *two* pipelines. In sharp contrast, Zaidi teaches a system that creates *delay* between two instructions within the *same* pipeline to thereby decrease or eliminate any parallelism had existed within the single pipeline system. Thus, Grochowski actually teaches away from Zaidi, and one skilled in the art would not be motivated to combine aspects of the Grochowski system with Zaidi.

Claim 4 claims using a combination of instructions in combination with a predetermined condition to enable the logic sequencer. For reasons discussed in regards to

Claim 3, the use of the combination of instructions is not taught or suggested by the cited references. In reference to the use of the "predetermined condition", the Examiner states that this "condition" is taught by the very existence of the instruction combination. (Office Action page 12 last full paragraph.) Applicants' Representative disagrees. If the instruction

5   combination and the predetermined condition were one in the same, Claim 4 would not make sense because the "only if" condition would always be satisfied. Additionally, Applicants' Specification clearly discusses the use of a predetermined condition that is *in addition* to the instruction combination. (See, for example, page 29 first full paragraph and page 33, line 26 through page 34, line 11 in reference to Figure 16.) This use of an *additional* condition in

10   *conjunction with* an instruction combination is not taught or suggested by any of the cited combination of references, and Claim 4 is allowable over this rejection for this additional reason.

Claim 7 relates to the aspect of Applicants' invention wherein one or more instructions are assigned a compare value. If these compare values have a predetermined

15   relationship, de-piping occurs. (Applicants' Specification pages 27 and 28.) The Examiner states that this is taught by Grochowski. (Office Action page 13 first full paragraph.) Grochowski teaches detecting a pair of instructions by determining that no register dependencies exist between the instruction pair, and further by determining that the instructions belong to a subset of instructions eligible for parallel execution. (Grochowski

20   column 4 lines 30-32.) Grochowski appears to detect dependencies by using a decoder that identifies the destination register for the first instruction, then determining whether the second instruction uses any resources needed by the first instruction. (Grochowski column 4 lines 32-37.) Thus, Grochowski does not teach assigning compare values to the instructions, then using those compare values to determine whether an instruction combination exists, as

25   claimed by Applicants' Claim 7. Further, as discussed above, Grochowski teaches a system that initiates *more*, not *less*, parallelism when any type of instruction combination is detected. Thus, Grochowski actually teaches away from Applicants' invention. For at least these additional reasons, Claim 7 is not taught or suggested by the cited combination of references, and this rejection should be withdrawn.

30   Claim 15 depends from Claim 11 and is allowable over this rejection for the reasons discussed above in reference to Claim 11. Claim 15 further includes aspects of Applicants' invention that are similar to those discussed above with respect to Claim 7. For at least the

additional reasons discussed above with respect to Claim 7, Claim 15 is allowable over this rejection.

Claims 21 and 22 depend from Claim 19, and are allowable for the reasons discussed above in regards to this Claim. Claim 21 further includes aspects of Applicants' invention that are similar to those discussed above in reference to Claim 3. For the additional reasons discussed above with respect to Claim 3, Claim 21 is allowable over this rejection. Claim 22 includes aspects of Applicants' invention that are similar to those discussed above in reference to Claim 4, and is allowable over this rejection for the additional reasons discussed above in reference to this Claim.

To summarize, Claims 3, 4, 7, 15, 21, and 22 describe aspects not taught or suggested by the cited combination of references, and these Claims are allowable over the current rejection, which is improper and should be withdrawn.

8. Applicants' Representative appreciatively acknowledges the indication of allowable subject matter in Claims 8-10 and 16-18. In view of the amendments to the Claims set forth above, and the arguments presented herein, it is respectfully submitted these Claims are in condition for allowance as presently presented.

9. The prior art made of record and not relied upon has been reviewed and is considered to be of general interest only.

CONCLUSION

Claims 1-22 were pending in the subject case. In the Office Action, Claims 1-7, 11-15, and 19-22 were rejected, and 8-10 and 16-18 were objected to. Claims 1-22 remain

5   pending. Claims 1, and ~~11-18~~ [11-15 and 18] were amended, and the remaining Claims are as previously presented. In view of the amendments to the Claims and the arguments set forth herein, it is respectfully submitted the Claims are in condition for allowance. If the Examiner has questions or concerns regarding this response, a call to the undersigned is encouraged and welcomed.

Respectfully submitted,

Beth L. McMahon   08/02/2002
Beth L. McMahon
Attorney for Applicants
Reg. No. 41,987
Tele No. (651) 635-7893

Unisys Corporation
M.S. 4773
P.O. Box 64942
St. Paul, MN 55164-0942

BLM/clk

15

<u>**VERSION WITH MARKINGS TO SHOW CHANGES MADE**</u>

Please change the title of the Application to <u>SYSTEM AND METHOD FOR CONTROLLING THE ENTRY OF INSTRUCTIONS INTO A PIPELINE OF AN INSTRUCTION PROCESSOR</u> frcm [PIPELINE DEPTH CONTROLLER FOR AN INSTRUCTION PROCESSOR].

IN THE SPECIFICATION:

Please amend the Abstract of the Invention on Page 44 as follows:

A programmable pipeline depth controller is provided to control the number of instructions that begins execution within an instruction pipeline of an instruction processor within a predetermined period of time. The pipeline depth controller of the present invention includes a logic sequencer responsive to a programmable count value. Upon being enabled, the logic sequencer generates a pipeline control signal to selectively delay the entry of some instructions into the instruction pipeline[. As a result,] <u>so that</u> the number of instructions that begins execution within the instruction pipeline during the predetermined period of time following the enabling of the logic sequencer is equal to the count value. [The flow of instructions through the instruction pipeline may be adjusted by re-programming the count value. Various modes of operation are provided for the pipeline depth controller. According to one mode, the pipeline depth controller is enabled to repeatedly generate the pipeline control signal in response to the selected count value. As a result, the number of instructions entering the pipeline during any period of time that is equal to the predetermined period of time is dictated by the count value. A second mode of operation is provided to enable the pipeline control signal to be generated in response to the entry of any of one or more selected instructions into the instruction pipeline. When one of the selected instructions enters the pipeline, the pipeline depth controller is enabled and is provided with a respective count value. During the predetermined period of time after the pipeline depth controller is enabled, the logic sequencer limits the number of instructions that enters the instruction pipeline to that number dictated by the count value. After the predetermined period of time elapses, the pipeline depth controller is disabled and the instruction pipeline continues execution in default mode. According to yet another mode of operation, the pipeline depth controller is enabled when any of one or more selected

combinations of instructions enters the pipeline. Each defined instruction combination may be associated with a respective count value. In this instance, the logic sequencer asserts control in a manner similar to that described above with respect to the entry of a single selected instruction into the pipeline. Still another mode allows the pipeline depth controller to be conditioned such that pipeline control is asserted only when a particular instruction combination enters the instruction pipeline while a particular system condition, such as a predetermined error, is occurring.]